(54) Title: MEDIA CODING FOR LOSS RECOVERY WITH REMOTELY PREDICTED DATA UNITS
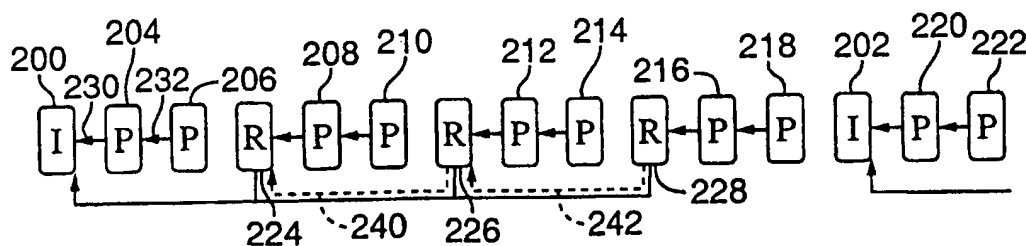


I — INDEPENDENT DATA UNIT
P — PREDICTED DATA UNIT
R — REMOTELY PREDICTED DATA UNIT

(57) Abstract

An improved loss recovery method for coding streaming media classifies each data unit in the media stream as an independent data unit (I unit), a remotely predicted unit (R unit) or a predicted data unit (P unit). Each of these units is organized into independent segments having an I unit, multiple P units and R units interspersed among the P units. The beginning of each segment is the start of a random access point, while each R unit provides a loss recovery point that can be placed independently of the I unit. This approach separates the random access point from the loss recovery points provided by the R units, and makes the stream more impervious to data losses without substantially impacting coding efficiency. The most important data units are transmitted with the most reliability to ensure that the majority of the data received by the client is usable. The I units are the least sensitive to transmission losses because they are coded using only their own data. While they provide the best coding efficiency, the P units are the most sensitive to data loss because the loss of one P unit renders useless all of the P units that depend on it. The remotely predicted units are dependent on the I unit, or in an alternative implementation, or another R unit.

- 1 -

## MEDIA CODING FOR LOSS RECOVERY WITH REMOTELY
## PREDICTED DATA UNITS

### FIELD OF THE INVENTION

5        The invention relates to the coding of streaming media, and in particular, relates to loss recovery in streaming media applications.

### BACKGROUND OF THE INVENTION

Streaming media is a form of data transfer typically used for multimedia content such as video,
10      audio, or graphics, in which a transmitter sends a stream of data so that a receiver can display or playback the content in real time. When multimedia content is streamed over a communication channel, such as a computer network, playback of the content becomes very sensitive to transmission delay and data loss. If the data does not arrive reliably or bandwidth on the client falls below an acceptable minimum, playback of the content is either delayed or discontinued. The rate of data transfer (e.g., the bit rate) required to
15      achieve realistic output at the receiver depends on the type and size of the media being transmitted.

In a typical application of streaming media, a server transmits one or more types of media content to a client. Streaming media is becoming more prevalent on the World Wide Web, where server computers deliver streaming media in the form of network data packets over the Internet to client computers. While multimedia data transfer over computer networks is a primary application of streaming
20      media, it is also used in other applications such as telecommunications and broadcasting. In each of these applications, the transmitter sends a stream of data to the receiver (e.g., the client or clients) over a communication channel. The amount of data a channel can transmit over a fixed period of time is referred to as its bandwidth. Regardless of the communication medium, the bandwidth is usually a limited resource, forcing a trade-off between transmission time and the quality of the media playback at
25      the client. The quality of playback for streaming media is dependent on the amount of bandwidth that can be allocated to that media. In typical applications, a media stream must share a communication channel with other consumers of the bandwidth, and as such, the constraints on bandwidth place limits on the quality of the playback of streaming media.

One way to achieve higher quality output for a given bandwidth is to reduce the size of the
30      streaming media through data compression. At a general level, streaming media of a particular media type can be thought of as a sequential stream of data units. Each data unit in the stream may correspond to a discrete time sample or spatial sample. For example, in video applications, each frame in a video sequence corresponds to a data unit. In order to compress the media with maximum efficiency, an encoder conditionally codes each data unit based on a data unit that will be transmitted to the client
35      before the current unit. This form of encoding is typically called prediction because many of the data units are predicted from a previously transmitted data unit.

-2-

In a typical prediction scheme, each predicted data unit is predicted from the neighboring data unit in the temporal or spatial domain. Rather than encoding the data unit, the encoder uses the neighboring data unit to predict the signal represented in the current unit and then only encodes the difference between the current data unit and the prediction of it. This approach can improve the coding efficiency tremendously, especially in applications where there is a strong correlation among adjacent data units in the stream. However, this approach also has the drawback that a lost data unit will not only lose its own data, but will also render useless all subsequent data units that depend on it. In addition, where a stream of data is converted into a stream of units each dependent upon an adjacent unit, there is no way to provide a random access point in the middle of the stream. As a result, playback must always start from the beginning of the stream.

In order to solve these problems, conventional prediction schemes typically sacrifice some compression efficiency by breaking the stream into segments, with the beginning of each segment coded independently from the rest of the stream. To illustrate this point, consider the typical dependency graph of data units of a media stream shown in Fig. 1.

The dependency graph in Figure 1 shows the data units in the order that they are located in the input data stream. From left to right, the data units represent an ordered sequence of data units in streaming media. In video coding applications, for example, each of these data units corresponds to a video frame that is encoded, and then transmitted to a receiver for playback.

Conventional prediction schemes classify the data units in the stream as either independent data units (shown marked with the letter I, e.g., 100, 102, and 104) or predicted data units (shown marked with the letter P, e.g., 106-128). The I units are independent in the sense that they are encoded using only information from the data unit itself. The predicted units are predicted based on the similarity of the signal or coding parameters between data units. As such they are dependent on the preceding data unit, as reflected by the arrows indicating the dependency relationship between adjacent data units (e.g., dependency arrows 130, 132, 134, and 136).

Because independent units are encoded much less efficiently than predicted units, they need to be placed as far apart as possible to improve coding efficiency. However, this causes a trade-off between coding efficiency, on the one hand, and data recovery and random access on the other. If a data unit is lost, the predicted units that depend on it are rendered useless. Therefore, independent data units need to be placed closer together to improve data recovery at the sacrifice of coding efficiency. As the independent units are placed closer together, coding efficiency decreases and at some point, the available bandwidth is exceeded. When the bandwidth is exceeded, the quality of the playback of streaming media suffers excessive degradation because the given bandwidth cannot maintain adequate quality with such poor coding efficiency.

Another drawback of the scheme shown in Fig. 1 is that the data recovery points must coincide with the random access points. Even if the need for random access does not force I units closer together,

the need for improved data recovery may anyway. As such, the coding scheme lacks the flexibility to treat data recovery and random access separately.

## SUMMARY OF THE INVENTION

5      The invention provides a coding method for streaming media that uses remote prediction to enhance loss recovery. Remote prediction refers to a prediction-based coding method for streaming media in which selected data units are classified as remotely predicted units. The coding method improves loss recovery by using remotely predicted units as loss recovery points and locating them independently of random access points in the data stream. The remotely predicted units improve loss
10     recovery because they depend only on one or a limited number of units located at a remote location in the encoded data stream. As a result they are less sensitive to data loss than a conventional predicted unit, which often depends on multiple data units. The remotely predicted units can be inserted closer together than independent units without substantially decreasing coding efficiency because they have a much higher coding efficiency than independent data units.

15     One aspect of the invention is a process for classifying the data units in streaming media to enhance loss recovery without significantly decreasing coding efficiency. Typically performed in the encoder, this process classifies data units as independent units (I units), predicted units (P units), or remotely predicted units (R units). Operating on an input stream of data units (e.g., a sequence of video frames), the process groups contiguous sequences of units into independent segments and classifies the
20     data units within each segment so that it includes an I unit, followed by P units, and one or more R units. The R units provide improved loss recovery because they depend only on the I unit in the segment, or, alternatively, another R unit. In addition, they are encoded more efficiently than I units because they are predicted from another data unit, as opposed to being coded solely from the unit's own data.

To support remote prediction, an encoder implementation classifies the data units as either I, P,
25     or R type units, and then encodes each differently according to their type. The encoder predicts both the P and R type units from a reference unit, but the reference unit is usually different for these data types. In particular, the R unit is predicted from the I unit in the segment, or alternatively, from another R unit. The P units are predicted from an adjacent unit in the stream (e.g., the immediately preceding data unit).

To support both forms of prediction at the same time, the encoder allocates two memory spaces, one each
30     for the reference units used to predict R and P units. In this encoding scheme, the independent segment typically starts with an I unit, which is followed by multiple P units, each dependent on the immediately preceding unit. R units are interspersed in the segment as needed to provide data recovery, while staying within bandwidth constraints.

In the decoder implementation, the decoder identifies the type of data unit, usually based on an
35     overhead parameter embedded in the bit stream, and then decodes the data unit accordingly. Like the encoder, the decoder allocates two memory spaces for storing reference data units, one used to

reconstruct the R type units and another to reconstruct P type data units. When the decoder identifies a data unit as an R unit, it reconstructs the original data unit using the I unit for the current segment, or alternatively, a previous R unit. When the decoder identifies a data unit as a P unit, it reconstructs the original data unit using the immediately preceding data unit, which has been previously reconstructed.

A variety of alternative implementations are possible. In particular, the data units, and specifically the R type units, can be classified dynamically based on some criteria derived or provided at run-time, or can be inserted based on a predetermined spacing of R units relative to the P units in a segment. Also, the I, P, and R units can be prioritized for transfer to improve error recovery and make the transmission more robust with respect to data losses. In particular, the data units are preferably prioritized so that I units are transferred with the most reliability, R units the second most reliability, and P units the least.

Further features and advantages of the invention will become apparent with reference to the following detailed description and accompanying drawings.

## BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a diagram of a typical dependency graph for a conventionally coded media stream.

Figs. 2A-B are diagrams illustrating dependency graphs produced by possible implementations of the invention.

Fig. 3 is a block diagram illustrating an encoder that employs the prediction-based coding scheme of the invention.

Fig. 4 is a block diagram of a decoder that is compatible with the prediction-based coding scheme shown in Fig. 3.

Fig. 5 is a block diagram of an object-based video encoder that incorporates the prediction-based coding scheme of the invention.

Fig. 6 is a block diagram of an object-based video decoder that is compatible with the prediction-based coding scheme shown in Fig. 5.

Fig. 7 is a block diagram of a computer system used to execute a software implementation of the invention.

## DETAILED DESCRIPTION

### Introduction

The following sections provide a detailed description of a multimedia coding scheme that uses a remotely predicted data unit to enhance loss recovery. The first section begins with an overview of an encoder and decoder implementation. The next section then provides further detail regarding an implementation of the encoder and decoder for video coding applications. Subsequent sections describe

- 5 -

methods for classifying the type of data units in an encoded bit stream and prioritizing the transfer of encoded units based on the data unit type.

　　　　To illustrate the concept of a remotely predicted data unit, it is helpful to consider a dependency graph of an encoded data stream that uses remotely predicted units. Fig. 2 is a diagram illustrating an example of a dependency graph of a data stream with remotely predicted units. For ease of comparison, it uses the same notation as used for the dependency graph shown in Fig. 1. The coding scheme shown in Fig. 1 places the independent data units approximately five to ten seconds apart for the purposes of loss recovery and random access. However, ideally, it would be preferable to place loss recovery points closer together, e.g., one or two seconds apart, to minimize the impact of a lost data unit. Also, it would be preferable to make the random access points independent of the loss recovery points because independent data units are coded much less efficiently than predicted data units.

　　　　If loss recovery and random access are considered independently, the independent data units can be placed further apart, e.g., ten to fifteen seconds or more, because sufficient random access can still be achieved even when the independent data units are spread further apart. Using remotely predicted data units as loss recovery points, a prediction-based coding scheme can place the loss recovery points nearly independently from the random access points. Fig. 2A shows a dependency graph of an encoded media stream, including independent data units (I, marked with reference numbers 200,202), predicted data units (P, marked with reference numbers 204-222), and remotely predicted data units (R, marked with reference numbers 224-228).

　　　　The arrows (e.g., 230, 232) interconnecting the data units in the stream illustrate the dependency relationships among each of the data units. The dependency relationship means that the dependent data units are coded based on some signal attribute of the reference data unit or on the coding parameters of the reference data unit. The independent data units are the first units in independent segments of the stream. For example, the independent data unit labeled 200 is the first data unit of an independent segment spanning from data unit 200 to data unit 218. Also, independent data unit 202 is the start of another independent data segment. Each of the independent data units is encoded using only information from a single, corresponding data unit in the input stream. For example, in a video stream, the data units correspond to image frames, and the independent data units correspond to intra-coded frames, which are coded using solely the content of the corresponding image frame.

　　　　As reflected by the arrows (e.g., 230-232) in the dependency graph shown in Fig. 2A, each of the P units are encoded using the immediately adjacent data unit in the data stream. In a temporal stream of data, this dependency relationship means that each of the P units is encoded based on the immediately preceding data unit in the time domain. Depending on the correlation between adjacent units, each of the P units achieves some level of coding efficiency by only encoding the differences between adjacent data units in the series. Each P unit is dependent upon the immediately preceding data

-6-

unit in the sense that the immediately preceding data unit is needed to compute the differences used to encode it and is also used for reconstructing the original data unit in the decoding process.

The remotely predicted data units (224-228) shown in Fig. 2A are "remote" from the data units that they depend on in the sense that they are not dependent upon the immediately adjacent data unit in the data stream. Instead, they are each dependent upon a remote, non-adjacent data unit, which in this case is the independent data unit at the beginning of the segment. In particular, the encoder codes a data unit classified as an R unit by coding the differences between the data unit and the prediction derived from the independent data unit at the beginning of the segment. In some coding systems, a dependent data unit (R or P type) is dependent in the sense that the dependent data units are coded based on the coding parameters of the reference data unit (e.g., the coding parameters in a parametric model).

While the remotely predicted data units shown in the dependency graph of Fig. 2A are each dependent upon the independent data unit in their segment, it is also possible to make them dependent upon the immediately preceding R unit in the segment. The dashed lines 240, 242 in Fig. 2A show the new dependency relationships for this alternative approach. This alternative approach enhances the encoding efficiency of remotely predicted data units because the differences between R classified units interspersed throughout a segment are likely to be smaller than the differences between these R units and the first data unit in the segment. However, it provides less effective loss recovery since a lost R unit renders the remaining R units in the segment useless in this case.

In the dependency graph shown in Fig. 2A, the random access points still correspond to the placement of independent data units (200, 202), which are now placed further apart relative to the dependency graph of Fig. 1. The remotely predicted data units (224, 226, 228) are placed nearly independently from the independent data units and are placed closer together than the I units to provide better loss recovery. The remotely predicted data units improve loss recovery without significantly reducing coding efficiency because they can be coded more efficiently than independent data units. If more efficiency is desired, remotely predicted units can be predicted from each other, instead of from the independent data unit at the beginning of the segment as noted above.

Fig. 2B illustrates another dependency graph with multiple levels of remotely predicted units, each dependent on either the I unit of the current segment or another R unit. In the coding approach illustrated in Fig. 2B, the encoder classifies the incoming stream of data units into multiple layers of dependency. The encoder spaces the I units in roughly the same fashion as shown in Fig. 2B. It spaces multiple levels of R type units such that the frequency of occurrence (e.g., frame rate) of each level of R unit increases with each level. The first level R type unit, $R_1$, is inserted roughly half way between the data units in an independent segment that starts with an I unit. The next level, $R_2$, is inserted so as to double the frame rate of the R type units. Note that the first $R_2$ unit is dependent on the I unit, and then the remainder of the $R_2$ units are dependent on $R_1$ units. The third level, $R_3$, is inserted so as to double the

frame rate of R type units relative to the previous level, $R_2$.   Note that the $R_3$ type units are dependent on the immediately preceding I or R type unit in the input stream, whichever is closer.

Each R type unit forms the beginning of a sub-segment in which each of the higher level R type units in the sub-segment are dependent on the first unit in the sub-segment.  Fig. 2B shows an example of
5      the partitioning of these segments as L1, L2 and L3 in the second half of the independent segment (the same structure also exists on the first half).  The data units in each sub-segment are dependent on the first data unit in the sub-segment at that level.  For example, all the units in L1 are either directly or indirectly dependent on the $R_1$ unit (250) at the beginning of the segment.  The sub-segment L1 contains two sub-segments at the next level, L2.  One starts with the $R_1$ unit 250 and another with the R2 unit 252.  Each
10     L2 sub-segment further contains two L3 segments.  Each of the data units in each L3 sub-segment are dependent on the first data unit in the segment (250, 256, 252, and 258), which is classified as an n-1 or lower level type unit (where n is the level).  The dots represent P units that are interspersed among the R type units, each being dependent on the immediately previous data unit.  Fig. 2B represents only an example of one possible dependency graph; the actual changes in frame rate from one level to the next as
15     well as the number of levels can vary.

The coding approach shown in Fig. 2B is particularly useful in a transmission scheme where the encoded data units are prioritized.  In such a scheme, the I units in a contiguous, sequential portion of the input stream are transmitted first, followed by the $R_1$ units, then the $R_2$ units and finally the $R_3$ units.  The server repeats this prioritized transmission for each contiguous portion in sequential order.  For example,
20     if a portion is 10 seconds long, the server would send all I units in the first ten second segment, then the $R_1$ units, then the $R_2$ units, and so on.  Next, it would repeat this process for the second ten second segment.  The amount of data sent in each portion depends on the bandwidth available to the client. Because this approach adds latency to the process of re-building the data stream on the client, the size of the portion needs to be selected within some tolerable latency constraint on the client.

25     Each of the R type units provides a loss recovery point that is dependent on some other data unit, either R or I, which is transmitted with higher priority.  Thus, if some P type units or even some R type units are lost, the decoder on the receiver only needs to select the next higher level R or I type unit to recover from the loss.  This improves loss recovery since the next R type unit is typically a lot closer than the next I unit would be in a conventional loss recovery scheme.

30     To implement this scheme, the remotely predicted data units encode the level of hierarchy that they reside in, as well as an identifier of their reference unit.  The decoder uses the level of hierarchy to determine which data unit is the proper recovery point in the event of data losses, and it uses the identifier of the reference unit to determine which previously reconstructed data unit it will use to reconstruct the current data unit.  The identifier may be implicit in the level of hierarchy of the remotely predicted unit.
35     In other words, the decoder may select the immediately preceding R unit in a lower level of hierarchy as the reference unit for the current R type unit.  In this context, the phrase, "immediately preceding" refers

- 8 -

to the order of the data units in the reconstructed data stream, which corresponds to the original order of the data units at the input to the encoder. The identifier may also explicitly identify a previously decoded data unit (e.g., frame of video) by the number of the reference data unit or an offset identifying the location of the reference frame in the reconstructed sequence of data units.

5      Media coders that incorporate the above-approach can be designed to insert the remotely predicted units at fixed locations in the data stream, or can place them selectively based on some other criteria. For example, an encoder may allow the user to specify the location of the I and R units as an adjustable input parameter. In addition, the encoder may be programmed to insert R units to improve loss recovery in cases where the media content has higher priority or where data losses are expected. At run-

10     time, a real-time encoder can be programmed to determine when data losses are expected by querying the communication interface for the data rate or for a measure of the data loss incurred for previously transmitted data units.

To mitigate losses, the encoder can be adapted to prioritize the transmission of the different data unit types. Independent data units, under this scheme, have the highest priority, then R units, and finally

15     P units.

Each of these features is elaborated on below.

**Encoder Overview**

Fig. 3 is a block diagram illustrating an implementation of a prediction-based coder that employs

20     remotely predicted units to improve loss recovery. The encoder shown in Fig. 3 operates on an input stream of contiguous data units (e.g., 300-306) and produces a coded bit stream of the form shown in Fig. 2A or 2B. Before coding the input data stream, a data unit classifier 310 classifies each of the data units in the input sequence as an independent (I), predicted (P), or remotely predicted (R) data unit. As explained in further detail below, the classification of data units can be adjusted as a user specified

25     parameter, or can be determined automatically in the encoder based on the media content, or some indicator of data transmission reliability, such as bit rate, a count of lost data packets, prediction error, etc.

Once classified, a predictor module 312 computes the prediction unit 314 for each of the data units in the input stream that are classified as R or P units. In most applications, the predictor module

30     will use some form of estimation to model the changes between a data unit and the data unit it is dependent on. One concrete example of this form of prediction is motion estimation performed in video based coding to predict the motion of pixels in one video frame relative to a reference video frame. In some applications, the changes between a data unit and the reference data unit may be so small that no estimation is necessary. In these circumstances, the prediction unit 314 is simply the reference data unit.

35     The reference unit is the data unit that the current data unit being encoded is directly dependent on. The arrows connecting data units in Figs. 2A and 2B illustrate this direct dependency relationship.

If the predictor 312 uses a form of estimation to improve coding efficiency, it produces a set of prediction parameters 316 in addition to the prediction unit 314. These prediction parameters are used to estimate changes between the data unit being decoded and its reference data unit. As shown in the dependency graph of Fig. 2A for example, the reference data unit for each new data unit being decoded is

5 either the I unit at the beginning of the data stream or the immediately preceding data unit. The predictor module uses the I unit to predict the first P unit in the segment and for at least one of the R units in the segment. For all other P type units in the segment, the predictor module uses the immediately preceding data unit in the input stream. The process of computing the prediction unit 314, therefore, requires that there be at least two separate memories allocated: one for storing a reference unit for remote prediction

10 (memory unit 318) and one for storing a reference unit for local prediction of P type units, namely, the immediately preceding data unit (memory unit 320).

If the coding system is implemented to support multi-levels of R units, then the encoder will allocate another memory unit for each level of remotely predicted unit. For example, a first level of remotely predicted units, $R_1$, may be inserted between units classified as I type in the original input

15 stream. Each $R_1$ unit is dependent on the I unit in the segment. A second level of remotely predicted units may be inserted between units classified as $R_1$ such that each of these second level ($R_2$) units is dependent on the $R_1$ or I unit at the beginning of the sub-segment. This multi-level hierarchy may be repeated for each level by interspersing data units $R_n$ in between the data units classified in the immediately preceding level $R_{n-1}$ in the hierarchy, where n represents the level of hierarchy. The default

20 spacing of the units $R_n$ at each level is such that the $R_n$ units are approximately evenly distributed. However, the spacing of R type units may vary at each level as explained in more detail below. Since each unit $R_n$ is dependent on at least the level below it (e.g., level n-1), this scheme requires that a memory unit be allocated for each level.

To reduce coding errors, the predictor module 312 uses a reconstructed version of the reference

25 data unit for prediction rather than a data unit read directly from the input stream. The predictor 312 uses the reconstructed data unit for prediction (either from memory unit 318 or 320, depending on whether or not the prediction is remote). This aspect of the encoder is reflected in Fig. 3 by the data processing path that flows into the locally reconstructed unit 320 for P type units and the reconstructed unit 318 for R type units. Note that this processing path includes an encoding phase followed immediately by a decoding

30 phase to compute the reference data unit stored in memory units 318, 320. Similarly, the encoder includes an independent data unit encoder 324 to encode the independent data unit 328 at the beginning of the stream, and subsequently, an independent unit decoder 326 reconstructs the data unit before being using it as a reference for predicting R units (see memory unit 318).

After the prediction stage, the encoder computes the differences between the prediction unit 314

35 and the data unit that it is dependent on (the reference unit). For example, in video applications, the error calculator 330 computes the differences between each pixel in the predicted image and the corresponding

- 10 -

pixel in the reference image, which is either the reconstructed version of the first frame in the sequence or the reconstructed version of the immediately adjacent frame in the sequence. The output of the error calculator 330 is the error signal 332, which is a representation of the differences between the current unit and its predictor.

The data classifier can also use the magnitude of the prediction error to choose coding between R and P type units. In particular, if the prediction error for P units is getting relatively high, it is worthwhile to insert an R unit instead, since it provides better loss recovery without unduly sacrificing fidelity in this case. When the error gets too big for an R type unit, for example, the data classifier should start a new independent segment, starting with a new I type unit. In addition, the encoder may be adapted to compute the prediction error for both an R and P type unit, and then pick the unit type based on which one has the least prediction error.

There are variety of ways of implementing the inter-process communication between the error calculator/predictor module and the data classifier module. For example, the error calculator 330 or predictor 312 can be designed to notify the data classifier when a threshold is exceeded by signaling an event that the data classifier monitors. Alternatively, the data classifier can query the error calculator periodically (e.g., after processing some number of data units in the stream) to check the error level of the most recent R frame. This interaction is reflected in Fig. 3 as an arrow from the error calculator 330 to the data unit classifier 310. Finally, the data classifier may simply be an integral part of the predictor module that selects the type of data unit based on the magnitude of the prediction error relative to a threshold (e.g., for I units) or based on a comparison of the prediction error for P and R type units.

After the error is computed for the current data unit, a predicted unit encoder 334 encodes the error signal 332 and prediction parameters 316 for the unit. The output of this portion of the encoder is either an encoded P type unit 336 or an R type unit 338. Since both of these types of data units share substantially the same format, the same encoder can be used to encode both of them. An example of this type of encoder for video applications is a conventional DCT or wavelet encoder. In this example, the encoder might contain a transform coder, such as a DCT or wavelet coder, to transform the signal into the frequency domain. Next, the encoder would quantize the frequency coefficients, and then perform some form of entropy coding such as Huffman or arithmetic coding on the quantized coefficients.

In most applications, the predicted unit encoder 334 will use some form of indicator such as a parameter or a flag that indicates the type of data unit. This parameter is used to signal the type of data unit being encoded so that the decoder can determine which previously reconstructed data unit to use as the reference data unit and which unit to use as a loss recovery point.

As noted above, the predicted data units are encoded using a reference unit that is already passed through the encoding and decoding phases. This process of reconstructing the reference unit for encoded data is illustrated, in part, as the predicted unit decoder 340. The predicted unit decoder 340 performs the reverse operation of the predicted unit encoder 334 to reconstruct the error signal 342. Continuing the

- 11 -

video coding example from above, the predicted unit decoder could be implemented with an inverse entropy coder to convert a stream of variable length codes into quantization indicies, followed by an inverse quantization process and an inverse transform coding method to transform the coefficients back into the spatial domain. In this case, the error signal 342 is different than the error signal 332 from the

5    error calculator 330 because the coding method is a lossy process. After reconstructing this error signal 342, the encoder adds (344) the error signal to the prediction unit 314 to compute the reconstructed unit and stores it in memory unit 318 or 320, depending on whether it is located at a reference point for dependent R or P units. As noted above, the encoder may have multiple different memory units to buffer reference units for R type data units if the encoder is designed to support multi-layered R type coding.

10    The encoder encodes independent data units using only the data from the single, corresponding data unit from the input stream. As shown in Fig. 3, an independent unit encoder 324 codes the independent unit for the current segment and produces an encoded I unit 326. For video applications, examples of this type of encoder include texture coders such as a DCT or wavelet coder for encoding video frames. For audio applications, this type of encoder may comprise a frequency domain or

15    parametric coder, for encoding audio frames.

As the encoded I, P, and R units are generated by the respective encoders, a coding format module 354 then integrates these encoded units into the coding format. The coding format module 354 organizes each of the encoded I, P, and R units into a prioritized order for transmission as explained in more detail below.

20    Then depending on the application, the prioritized data may be further formatted into some form of packet or data stream that is specific to the communication channel being used to transmit the data. In Fig. 3, this final stage of data formatting is reflected as the channel interface 356. The channel interface 356 is responsible for placing the data in the form necessary to transmit it over the channel 358. For example, if the data is being transferred over a computer network, the channel interface is a network

25    adapter interface that takes the data from the coding format module 354 and organizes it in data packets compatible with the network communication channel 356. This network interface may be implemented using conventional network protocols such as TCP/IP, and a network driver, e.g. a software driver for a Ethernet network adapter.

In some implementations, the channel interface can provide a measure of the data transfer

30    reliability/available bandwidth that is useful to the data unit classifier 310 for determining where to insert R and I type data units in the stream. The arrow back to the data classifier from the channel interface represents the communication of this information back to the data classifier. The data classifier can get this information, for example, by querying the channel interface via a function call, which returns a parameter providing the desired information.

35    Fig. 4 is a diagram illustrating a decoder compatible with the encoder shown in Fig. 3. The decoder operates on an encoded media stream from a channel interface 402. The channel interface 402 is

responsible for sending and receiving data from a communication channel 404, and is identical to or at least compatible with the channel interface on the transmitter side. For example, in cases where the communication channel is a computer network, the channel interface is a network interface designed to un-package network data packets and place them in a form that they were originally in when submitted to the channel interface on the transmitter side.

The decoding process begins by reading the unpackaged data from the channel interface and breaking it into three different types of encoded data units: independent data units 406, predicted data units 408, and remotely predicted data units 410. The decode format module 405, shown in Fig. 4, is the component responsible for parsing the encoded input stream and sending each data unit to the appropriate decoder. A predicted unit decoder 412 is responsible for decoding both the predicted data unit 408 and the remotely predicted data units 410. The encoding format is similar for both R and P types as explained above, except that the reference unit used for prediction is chosen differently. In addition to decoding the error signal, the predicted unit decoder identifies and decodes the prediction parameters 416 from the encoded data.

The predictor 418 uses the prediction parameters and the previously decoded reference unit to compute the prediction unit 420 for the current data unit being decoded. The predictor 418 computes the prediction unit differently depending on whether the data unit being decoded is an R or P unit. In the case of an R unit, the predictor 418 selects the reference unit for the R unit stored in memory unit 422. For a first-level R unit, $R_1$, this reference unit is the reconstructed I unit for the segment. The decoder allocates sufficient memory units to buffer the reference units for each level of remotely predicted units. For simplicity, the diagram only shows a single memory unit 422, but the implementation may allocate more memory units, depending on the number of levels in the remote prediction scheme. For a P unit, the predictor 418 selects the immediately preceding data unit, which is stored separately from the reference unit for R type units (shown as the reconstructed unit 424 for P type units in Fig. 4). Note, however, a special case exists in that the P unit that immediately follows the first data unit in the segment or the first R type unit in a sub-segment will be predicted from the reference data unit 422, because in this case, it is the immediately preceding data unit. For subsequent data units in the segment, the predictor selects the immediate preceding data unit for P units and selects the R reference unit for remotely predicted units.

After computing the prediction unit 420 for the current data unit being decoded, the decoder combines the error signal 414 with the prediction unit 420 to reconstruct the current data unit. It stores the results in memory 424 allocated separately from the memory allocated for the decoded reference unit 422 for R units because both the decoded R reference unit and the immediately preceding data unit are necessary to decode the segment. With the decoding of each P and R unit, the decoder transfers the preceding data unit currently stored in memory 424 to a module for reconstructing the output (e.g., reconstruct output module 426). It then replaces the reference unit stored in memory 424 with the most recently decoded unit.

- 13 -

The decoder decodes the independent data units using the inverse of the steps performed in the independent unit encoder shown in Fig. 3. These operations are reflected generally as the independent unit decoder module 430 shown in Fig. 4. The decoded independent data unit is stored in memory 422, and as described above, is used to decode the first level R type units in the segment.

5          The decoder transfers a copy of the independent data unit at the beginning of the stream to the module for reconstructing the output, and replaces the independent data unit in memory 422 with each new independent segment in the stream.

Figs. 3 and 4 provide a general implementation of a codec that employs remote prediction to improve loss recovery with better coding efficiency. This codec can be incorporated into

10        encoders/decoders for variety of media streams that employ prediction based coding. It is particularly well suited for video coding, and as such, a more detailed description of a video encoder and decoder follow below.

**Video Coding Applications**

15        In video coding applications, the stream of data units correspond to a sequence of temporally spaced apart images, typically called frames. Presently, there are two primary classes of video coders: frame-based coders and object based coders. In both types of coders, the temporal sequence is comprised of discrete time samples coinciding with each frame, and the data at each time sample is an image. Frame-based and object based coders differ in that object based coders operate on arbitrary shaped

20        objects in the video scene, whereas frame-based coders operate on the entire frame. Object based coders attempt to exploit the coherence of objects in the frames by coding these objects separately. This may involve simply coding a foreground object, such as talking head, separately from a mostly static background. It also may involve more complicated scenes with multiple video objects. An object based coder can be reduced to a frame based coder by treating each frame as a single object. Examples of

25        video coding standards for frame based video encoding include MPEG-2 and H262-H263. The coding standard MPEG-4 addresses object based video coding.

While remote prediction may be used in both frame based and object based video coding, the following example illustrates remote prediction in the context of object based video coding.

Fig. 5 is a block diagram illustrating an implementation of an object-based video encoder that

30        supports remote prediction. The encoder modules shown in Fig. 5 operate on an input stream of frames, after a pre-processing stage has segmented the frames into objects and has classified each object as an I, P, or R unit. The input 530 to the encoder includes images representing the video objects in each frame, the shape of each video object and bounding rectangles. The shape information is available before the encoder codes texture or motion data. Frame-based coding differs in that the entire frame is coded

35        without shape information, and the input 530 simply consists of a series of image frames, each classified as R, I, or P frames.

- 14 -

The shape coding module 532 reads the definition of an object including its bounding rectangle and extends the bounding rectangle to integer multiples of fixed sized pixel blocks called "macroblocks." The shape information for an object comprises a mask or "alpha plane." The shape coding module 532 reads this mask and compresses it, using for example, a conventional chain coding method to encode the

5      contour of the object.

Motion estimation module 534 performs motion estimation between an object in the current frame and its reference object. This reference object is the same object in either the immediately preceding frame for P type data, or the same object in the first frame of the current segment, for R type data. The encoder stores the immediately preceding object as the reconstructed image 536, and stores the

10     object from the first frame in the segment as part of the I-frame image memory 550. The motion estimation module 534 reads the appropriate object from either memory space 536 or 550 based on a parameter that classifies the type of the object as either an R or P type. Using the stored image object and its bounding rectangle, the motion estimation module computes motion estimation data used to predict the motion of an object from the current frame to the reference frame.

15     In the implementation, the motion estimation module 534 searches for the most similar macroblock in the reference image for each macroblock in the current image to compute a motion vector for each macroblock. The specific format of the motion vector from the motion estimation module 534 can vary depending on the motion estimation method used. In the implementation, there is a motion vector for each macroblock, which is consistent with current MPEG and H26X formats. However, it is

20     also possible to use different types of motion models, which may or may not operate on fixed sized blocks. For example, a global motion model could be used to estimate the motion of the entire object, from the current frame to the reference frame. A local motion model could be used to estimate the motion of sub-regions within the object such as fixed blocks or variable sized polygons. The particular form of motion estimation is not critical to the invention.

25     Returning again to Fig. 5, the motion compensation module 538 reads the motion vectors computed by the motion estimation module and the previously reconstructed image 536 or the I frame image 550 and computes a predicted image for the current frame. Each pixel in the predicted image is constructed by using the motion vector for the macroblock that it resides in to find the corresponding pixel in the previously reconstructed image 536, for P type data, or the I frame image, for R type data.

30     The encoder then finds the difference between the image sample values in the input image block as specified in the input 530 and the corresponding sample values in the predicted image block as computed in the motion compensation module 538 to determine the error signal for each macroblock.

Texture coding module 540 compresses blocks of error signals for inter-frame coded objects, namely, R and P type objects. In addition, it compresses blocks of image sample values for the object

35     from the input data stream 530 for intra-frame coded objects, namely the I type objects. The feedback path 542 from the texture coding module 540 represents blocks of signals that were texture coded and

then texture decoded, including the reconstructed error signal and I-frame object. The encoder uses the error signal blocks along with the predicted image blocks from the motion compensation module to compute the reconstructed image 536. The reconstructed image memory 536 shown in Fig. 5 includes multiple memory units to reflect that more than one reference image may be needed to support multiple

5 levels of R type units, the current reconstructed I type unit, and the current P type unit.

The texture coding module 540 codes intra-frame and error signal data for an object using any of a variety of still image compression techniques. Example compression techniques include DCT, wavelet, as well as other conventional image compression methods.

The bit stream of the compressed video sequence includes the shape, motion and texture coded

10 information from the shape coding, motion estimation, and texture coding modules. In addition, it includes overhead parameters that identify the type of coding used at the frame, object, and macroblock levels. Depending on the implementation, the parameter designating the data type as I, R, or P can be encoded at any of these levels. In the implementation, the encoder places a flag at the macroblock level to indicate which type of data is encoded in the macroblock. Multiplexer 544 combines and formats the

15 shape, motion, texture, and overhead parameters into the proper syntax and outputs it to the buffer 546. The multiplexer may perform additional coding of the parameters. For example, overhead parameters and the motion vectors may be entropy coded, using a conventional entropy coding technique such as Huffman or arithmetic coding.

While the encoder can be implemented in hardware or software, it is most likely implemented in

20 software. In a software implementation, the modules in the encoder represent software instructions stored in memory of a computer and executed in the processor, and the video data stored in memory. A software encoder can be stored and distributed on a variety of conventional computer readable media. In hardware implementations, the encoder modules are implemented in digital logic, preferably in an integrated circuit. Some of the encoder functions can be optimized in special-purpose digital logic

25 devices in a computer peripheral to off-load the processing burden from a host computer.

Fig. 6 is a block diagram illustrating a decoder for an object-based video coding method. A demultiplexer 660 receives a bit stream 662 representing a compressed video sequence and separates shapes, motion and texture encoded data on an object by object basis. The demultiplexer also includes a mode selector to indicate whether the macroblocks are I, P, or R type.

30 Shape decoding module 664 decodes the shape or contour for the current object being processed. To accomplish this, it employs a shape decoder that implements the inverse of the shape encoding method used in the encoder of Fig. 5. The resulting shape data is a mask, such as a binary alpha plane or gray scale alpha plane representing the shape of the object.

The motion decoding module 666 decodes the motion information in the bit stream. The

35 decoded motion information includes the motion vectors for each macroblock that are reconstructed from entropy codes in the incoming bit stream. The motion decoding module 666 provides this motion

- 16 -

information to the motion compensation module 668. The motion compensation module 668 uses the motion vectors to find predicted image samples in the previously reconstructed object data 670 for P type macroblocks and higher level R type macroblocks, or the I-frame object memory 680 for first level R type macroblocks.

5      The texture decoding module 674 decodes error signals for inter-frame coded texture data (both R and P type) and an array of color values for intra-frame texture data (I type data) and passes this information to a module 672 for computing and accumulating the reconstructed image. It specifically stores the I frame data objects in separate memory 680 so that the motion compensation module can compute the predicted object for first level R type objects using the I frame's object rather than the

10     immediately preceding frame's object. For inter-frame coded objects, including both R and P type data, this module 672 applies the error signal data to the predicted image output from the motion compensation module 668 to compute the reconstructed object for the current frame. For intra-frame coded objects, the texture decoding module 674 decodes the image sample values for the object and places the reconstructed object in the reconstructed object module 672 and I-frame object memory 680. Previously reconstructed

15     objects are temporarily stored in object memory 670 and are used to construct the object for other frames.

Like the encoder, the decoder can be implemented in hardware, software or a combination of both. In software implementations, the modules in the decoder are software instructions stored in memory of a computer and executed by the processor, and video data stored in memory. A software decoder can be stored and distributed on a variety of conventional computer readable media. In hardware

20     implementations, the decoder modules are implemented in digital logic, preferably in an integrated circuit. Some of the decoder functions can be optimized in special-purpose digital logic devices in a computer peripheral to off-load the processing burden from a host computer.

**Classifying R Type Data Units**

In a coding system that supports remotely predicted data units, the encoder preferably classifies

25     each data unit as an I, R, or P type unit for encoding. There are a variety of ways of implementing the classification scheme. One approach is to implement the encoder so that it always uses the same spacing of I, R, and P data units. Another approach is to allow the user to specify the spacing by making the spacing of each type a user adjustable parameter. Yet another approach is to program the encoder to classify the data units in the stream adaptively or dynamically based on some criteria determined during

30     the encoding process. A scheme for adaptively classifying data units refers to a process in which the encoder selects the data type by adapting it to some characteristic of the media content or some other criteria relating to the coding of the media content or its transmission. The term "dynamic" is similar to "adaptive" and is meant to apply to implementations where data unit type is selected at run-time (during operation) of the encoder or during transmission of the stream.

35     The classification of data unit types may be dynamic or predetermined based on the media content type. The encoder may determine the spacing of I and R units in the stream based on a

- 17 -

predetermined relationship between the media content type and the spacing of I and R units. The encoder, for example, may be programmed to use a default spacing of R units for all video sequences of a certain type. In particular, the encoder might use a first default spacing of R units for low motion talking head video and a second, different spacing for high motion video scenes. The encoder may allow the user

5    to specify a different spacing for the content via a user-adjustable input parameter.

One form of dynamically classifying data units is to select the spacing of I, P and R type units in the stream based on a prediction error threshold. For example, the encoder can be programmed to start a new independent segment with an I type unit when the error signal from the prediction process exceeds a predetermined or user-adjustable threshold. In this case, the encoder may be programmed to classify

10   selected data units in between each I unit in the sequential stream as R type units according to a predetermined or user-adjustable spacing. The encoder would classify the remaining data units as P type units.

Another measure of data transfer reliability is an indicator of the quantity of data being lost for previously transferred data packets over the communication channel (such as network data packets). The

15   encoder could also obtain this information from the channel interface. In particular for computer networks, the network interface software would keep track of the dropped data packets. The encoder could then query the interface to request that it return the number of lost packets. The encoder would then calculate or adjust the spacing of R units for each independent segment based a predetermined relationship between the number of lost data packets over a period of time and the spacing of R units.

20   Another form of dynamically classifying data units is to select the spacing of R type units based on a measure of the fidelity of the output stream at the client. One approach is to use the magnitude of the prediction error as a measure of the fidelity. The magnitude of the prediction error can be used to select coding between R and P type data units on the server.

As demonstrated above, there are a number of ways to control the classification of the data unit

25   types. An implementation of the invention may use one or more of these methods, or some alternative method. Note, that in general, the R units may be classified independently from the I units. In other words, the spacing of the R units may be selected so as to be totally independent of the spacing of I units. Thus, the encoder can optimize the spacing of R units depending on a variety of criteria independent of the need for random access points.

30

**Prioritizing Data Units**

As introduced above, the encoder implementation may also prioritize the transfer of I, P, and R encoded data units with respect to the reliability of the data transfer. Preferably, the encoded stream should be prioritized such that the I unit has the highest reliability of being transferred, the R unit has the

35   same or lower reliability as the I unit, and the P unit has the same or lower probability as the R unit. Additionally, if the encoder implements a multi-layered R encoding scheme, the priority for each level $R_n$

- 18 -

would decrease. For example, units of type $R_1$, dependent on the I unit in the segment, would have the highest priority, units of type $R_2$, dependent on unit of type $R_1$ in the sub-segment, would have the next highest priority.

The implementation prioritizes data transfer by transmitting the higher priority data units first so
5   that there is more time to retransmit them if they are not received. Once prioritized, the encoded data units maintain that priority level on retransmission. In particular, all higher priority data units are transmitted, including retransmissions if necessary, before lower priority data units.

The implementation of the prioritizing scheme operates in conjunction with an underlying transport mechanism that supports prioritization of data transfer based on some pre-determined block of
10  data (e.g., a packet). The encoder instructs the transport mechanism to place higher priority data in higher priority data packets for transfer, and ensures that these higher priority packets are successfully transmitted before instructing the transport mechanism to send the next lower priority data unit types. This process of sending all data units of a certain type in prioritized fashion operates on a portion of the input data stream, such that all of the I type units in that portion are sent first, then all of the $R_1$ type units
15  in that portion, then all of the $R_2$ units in that portion, and so on. Preferably, the size of this portion is set so as not to add significant latency on the client side. For example, the size of the portion size may be set as 10 seconds worth of data units in a video stream. If the size of the portion becomes larger, the client incurs additional latency waiting for receipt of the portion, before it is able to starting reconstructing the stream.

20  This approach for prioritizing the transfer of the encoded stream helps to ensure that any losses will affect the P unit first, before affecting the R or I units. As a result, the client is more likely to receive the most useful information, namely the independent data units, which provide random access and data recovery, and the higher priority R type units, which provide additional data recovery points.

The prioritization scheme works in the case of a transfer between a server and a single client,
25  and in a multi-cast scenario where a server broadcasts the data packets to multiple clients. The extent to which the client uses the encoded units to reconstruct the output depends on its available transmission bandwidth. In the case of a multi-cast application, the client can subscribe to the number of priority levels that it can support based on its available bandwidth. Consider an example such as Fig. 2B where the data stream is encoded with multiple layers of R units, and the units in each layer are dependent on
30  the R unit in a lower level sent with higher priority. Also, consider the case where the frame rate of the R units designated in the input stream doubles with each level of hierarchy. In particular, the highest priority R units, $R_1$, are inserted so as to double the frame rate relative to the I units, the next highest priority R units, $R_2$, are inserted so as to double the frame rate relative to the $R_1$ units, and so on. Depending on the bandwidth available to the client, it can subscribe to however many levels it wants,
35  with each level providing better fidelity yet increasing the bandwidth requirements. By sending all units

- 19 -

of a given priority level in a portion of the stream first, this approach sacrifices the lower priority units to make sure that the higher priority units will be received.

**Brief Overview of a Computer System**

5        Figure 7 and the following discussion are intended to provide a brief, general description of a suitable computing environment in which the invention may be implemented. Although the invention or aspects of it may be implemented in a hardware device, the encoder and decoder described above are implemented in computer-executable instructions organized in program modules. The program modules include the routines, programs, objects, components, and data structures that perform the tasks and

10    implement the data types described above.

        While Fig. 7 shows a typical configuration of a desktop computer, the invention may be implemented in other computer system configurations, including hand-held devices, multiprocessor systems, microprocessor-based or programmable consumer electronics, minicomputers, mainframe computers, and the like. The invention may also be used in distributed computing environments where

15    tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote memory storage devices.

        Figure 7 illustrates an example of a computer system that serves as an operating environment for the invention. The computer system includes a personal computer 720, including a processing unit 721, a

20    system memory 722, and a system bus 723 that interconnects various system components including the system memory to the processing unit 721. The system bus may comprise any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using a bus architecture such as PCI, VESA, Microchannel (MCA), ISA and EISA, to name a few. The system memory includes read only memory (ROM) 724 and random access memory (RAM) 725. A basic

25    input/output system 726 (BIOS), containing the basic routines that help to transfer information between elements within the personal computer 720, such as during start-up, is stored in ROM 724. The personal computer 720 further includes a hard disk drive 727, a magnetic disk drive 728, e.g., to read from or write to a removable disk 729, and an optical disk drive 730, e.g., for reading a CD-ROM disk 731 or to read from or write to other optical media. The hard disk drive 727, magnetic disk drive 728, and optical disk

30    drive 730 are connected to the system bus 723 by a hard disk drive interface 732, a magnetic disk drive interface 733, and an optical drive interface 734, respectively. The drives and their associated computer-readable media provide nonvolatile storage of data, data structures, computer-executable instructions (program code such as dynamic link libraries, and executable files), etc. for the personal computer 720. Although the description of computer-readable media above refers to a hard disk, a removable magnetic

35    disk and a CD, it can also include other types of media that are readable by a computer, such as magnetic cassettes, flash memory cards, digital video disks, Bernoulli cartridges, and the like.

- 20 -

A number of program modules may be stored in the drives and RAM 725, including an operating system 735, one or more application programs 736, other program modules 737, and program data 738. A user may enter commands and information into the personal computer 720 through a keyboard 740 and pointing device, such as a mouse 742. Other input devices (not shown) may include a

5 microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit 721 through a serial port interface 746 that is coupled to the system bus, but may be connected by other interfaces, such as a parallel port, game port or a universal serial bus (USB). A monitor 747 or other type of display device is also connected to the system bus 723 via an interface, such as a display controller or video adapter 748. In addition to the monitor, personal

10 computers typically include other peripheral output devices (not shown), such as speakers and printers.

The personal computer 720 may operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 749. The remote computer 749 may be a server, a router, a peer device or other common network node, and typically includes many or all of the elements described relative to the personal computer 720, although only a memory storage device 750 has

15 been illustrated in Figure 7. The logical connections depicted in Figure 7 include a local area network (LAN) 751 and a wide area network (WAN) 752. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets and the Internet.

When used in a LAN networking environment, the personal computer 720 is connected to the local network 751 through a network interface or adapter 753. When used in a WAN networking

20 environment, the personal computer 720 typically includes a modem 754 or other means for establishing communications over the wide area network 752, such as the Internet. The modem 754, which may be internal or external, is connected to the system bus 723 via the serial port interface 746. In a networked environment, program modules depicted relative to the personal computer 720, or portions thereof, may be stored in the remote memory storage device. The network connections shown are merely examples

25 and other means of establishing a communications link between the computers may be used.


**Conclusion**

While the invention has been illustrated using a specific implementation as an example, the scope of the invention is not limited to the specific implementation described above. In particular, the use

30 of remote prediction scheme for improved error recovery is not limited to video, but instead, applies to other media streams as well. For example, the same techniques described above can be applied to audio, where the data units are frames of audio and there is a dependency relationship among the audio frames. In audio coders available today, an audio frame corresponds to a fixed number of PCM samples in the time domain. When an audio frame is made to be dependent on another audio frame, whether it is a

35 dependency between audio signals or audio coding parameters (e.g., the parameters in a parametric

- 21 -

coding model in speech), remote prediction can be used to improve error recovery in the manner described above. In particular one or more layers of R units may be inserted to improve loss recovery.

5   The data dependency graph of the encoded data stream may vary as well. For example, the independent data unit does not have to be at the beginning of an independent segment of data units. For example, the independent unit may be located in the middle of a temporal sequence of data units in an independent segment, rather than at the beginning of the segment. The other predicted units in the segment would be predicted directly or indirectly from the I unit. Note that the only restriction is that the I unit be available for decoding the other units in the segment that depend on it. This constraint can be satisfied by prioritizing the transfer of the data units such that the I unit is available for decoding the

10   dependent units.

As a practical matter, the I unit will likely be at the beginning of an ordered, temporal sequence of data units in an independent segment of the stream for most applications because the receiver usually will need to play it first before the other units in the segment. However, with the use of a prioritization scheme that re-arranges the data units for transmission, there is flexibility in the classification of the data

15   units as I, P, and R units.

Also, as noted above, the remotely predicted data units can use another R type unit as their reference, rather than the I unit in the segment. Rather than maintaining a copy of the I data unit, the coder/decoder would keep a copy of the previously reconstructed R unit in addition to the previously reconstructed P unit. The coding scheme described above could be extended to a multi-layered coding

20   scheme where there multiple levels of R type units. Each level may be dependent directly on the I unit in the segment, or on other R units, which in turn, are dependent on other R units or the I unit in the segment.

In view of the many possible implementations of the invention, it should be recognized that the implementation described above is only an example of the invention and should not be taken as a

25   limitation on the scope of the invention. Rather, the scope of the invention is defined by the following claims. We therefore claim as our invention all that comes within the scope and spirit of these claims.

- 22 -

We claim:

     1. A method for coding streaming media comprising a series of data units, the method

5    comprising:

     classifying each of the data units in the series as one of the following types of encoded data units: an independent unit, a predicted unit, and a remotely predicted unit, such that the data units in the series are organized into segments, and each segment has an independent data unit, two or more predicted units and at least one remotely predicted unit, wherein the independent data unit is a data recovery point

10   and a random access point in the series of data units, and the remotely predicted unit is a data recovery point in the series of data units that is classified independently from the random access point and is coded with more efficiency than the independent data unit;

     encoding each of the data units classified as an independent data unit in a compressed format using only information from the data unit;

15    encoding each of the data units classified as a predicted unit in a compressed format by encoding differences between the data unit and the immediately preceding data unit in the series; and

     encoding each of the data units classified as a remotely predicted unit in a compressed format by encoding differences between the data unit and the data unit classified as the independent unit in the segment.

20

     2.  The method of claim 1 including:

     encoding the series of data units as a sequence of encoded data units comprising contiguous segments, each contiguous segment starting with an encoded independent data unit, followed by predicted units that are each dependent on the immediately preceding data unit, and including at least one remotely

25   predicted unit inserted within a sequence of the predicted units;

     wherein the classifying step is performed dynamically while previously classified and encoded data units are being transmitted.

     3. The method of claim 2 wherein the classifying step includes dynamically

30   selecting a spacing of remotely predicted units in each of the contiguous segments as the encoded data units are being transmitted.

     4. The method of claim 3 wherein the spacing of remotely predicted units is determined dynamically based on a priority assigned to media content being transmitted.

35

5. The method of claim 3 wherein the spacing of remotely predicted units is determined dynamically based on data transmission rate.

6. The method of claim 3 wherein the spacing of remotely predicted units is dynamically determined based on a measure of data loss detected in previously transmitted data units.

7. The method of claim 1 further including;

encoding the series of data units as a sequence of encoded data units comprising contiguous segments, each contiguous segment starting with an encoded independent data unit, followed by predicted units that are dependent on the immediately preceding data unit, and including at least one remotely predicted unit inserted in a sequence of the predicted units such that the predicted unit immediately following the remotely predicted unit is predicted from the remotely predicted unit;

prioritizing encoded data units for transmission such that independent data units are transmitted with highest priority, remotely predicted units are transmitted with next highest priority, and predicted units are transmitted with lowest priority.

8. The method of claim 7 wherein the series of data units is divided into portions, and the data units in each portion are prioritized such that all independent data units in a portion are to be sent first, then all of the next highest priority data units, and finally all of the predicted units, subject to available bandwidth constraints.

9. The method of claim 1 wherein the data units are video frames.

10. The method of claim 1 wherein the data units are audio frames.

11. A computer readable medium having instructions for performing the steps of claim 1.

12. A method for decoding streaming media comprising a series of data units where the data units are arranged in segments, and each segment includes an independent data unit, two or more predicted units and at least one remotely predicted unit, the method comprising:

decoding an encoded bit stream to identify the data units in the series for decoding as one of the following types of coded data units: an independent unit, a predicted unit, and a remotely predicted unit;

for segments of independent data units in the series, decoding each of the segments, including:

decoding each of the data units classified as an independent data unit in a compressed format using only information from the data unit;

-24-

decoding each of the data units classified as a predicted unit by decoding differences between the data unit and the immediately preceding data unit in the series, and adding the differences with an immediately preceding data unit, which has been previously reconstructed and stored; and

decoding each of the data units classified as a remotely predicted unit by decoding differences between the data unit and the data unit classified as the independent unit in the segment, and adding the differences with the independent unit, which has been previously reconstructed and stored; and

in an event where a predicted unit is lost, performing loss recovery with a remotely predicted unit following the lost predicted unit;

wherein the independent data unit is a data recovery point and a random access point in the series of data units, and the remotely predicted unit is a data recovery point in the series of data units that is inserted at a location that is independent from the random access point and is coded with more efficiency than the independent data unit.

13. A computer readable medium having instructions for performing the steps of claim 12.

15. A method for classifying data units in a media stream for prediction-based coding, the method comprising:

reading an ordered sequence of data units in an input media stream;

classifying each of the data units in the series as one of the following types of encoded data units: an independent unit, a predicted unit, and a remotely predicted unit, such that the data units in the series are organized into segments, and each segment has an independent data unit, two or more predicted units and two or more remotely predicted units, wherein the independent data unit is a data recovery point and a random access point in the series of data units, and the remotely predicted units are data recovery points in the series of data units that are classified independently from the random access point and arc located closer together in the series of data units than the independent data units;

wherein each of the data units classified as an independent data unit is designated to be encoded using only information from the data unit;

wherein each of the data units classified as a predicted unit is designated to be predicted from an adjacent data unit in the series; and

wherein each of the data units classified as a remotely predicted unit is designated to be predicted from a remote, non adjacent data unit in the series, which is either another remotely predicted unit or an independent data unit.

16. The method of claim 15 wherein each segment corresponds to a segment in an ordered sequence of segments in the input media stream;

- 25 -

wherein each segment in the input media stream is a temporally ordered sequence of data units starting with a first data unit, and then followed by a temporally ordered sequence of data units;

wherein the classifying step classifies the first data unit in each segment as an independent data unit, and classifies each data unit following the first data unit in the segment as a predicted unit or a remotely predicted unit.

17. The method of claim 15 wherein the remotely predicted units are classified dynamically during transmission of previously encoded data units based on a measure of data transfer reliability.

18. The method of claim 15 wherein the remotely predicted units are classified based on a measure of available bandwidth for transferring encoded data units.

19. The method of claim 15 wherein the remotely predicted units are classified based a user adjustable input parameter.

20. The method of claim 19 wherein the user adjustable parameter indicates spacing of remotely predicted units in independent segments.

21. The method of claim 19 wherein the remotely predicted units are classified based on a user definable relationship between type of media content and spacing of the remotely predicted units in independent segments.

22. The method of claim 15 wherein the remotely predicted units are classified based on a parameter returned by a receiver of the data stream that plays a decoded version of the data stream, where the parameter provides a measure of fidelity of the playback of the data stream.

23. The method of claim 15 wherein the remotely predicted units are classified such that the series includes two or more levels of remotely predicted units, with a first level of remotely predicted units in a segment that are each directly dependent on an I unit of the segment, at least a second level of remotely predicted units in the segment, including a second level unit that is directly dependent on a unit in the first level.

24. A computer readable medium having instructions for performing the steps of claim 15.

25. A method for coding streaming media comprising a series of data units, the method comprising:

- 26 -

classifying each of the data units in the series as one of the following types of encoded data units: an independent unit, a predicted unit, and a remotely predicted unit, such that the data units in the series are organized into segments, and each segment has an independent data unit, two or more predicted units and at least one remotely predicted unit, wherein the independent data unit is a data recovery point

5      and a random access point in the series of data units, and the remotely predicted unit is a data recovery point in the series of data units that is classified independently from the random access point and is coded with more efficiency than the independent data unit;

encoding each of the data units classified as an independent data unit in a compressed format using only information from the data unit;

10     encoding each of the data units classified as a predicted unit in a compressed format by encoding differences between the data unit and an adjacent data unit in the series; and

encoding each of the data units classified as a remotely predicted unit in a compressed format by encoding differences between the data unit and a remote, non-adjacent data unit in the segment, selected as either the independent unit or another remotely predicted unit in the segment.

15

26.  A method for coding streaming media comprising a series of data units, the method comprising:

classifying each of the data units in the series as one of the following types of encoded data units: an independent unit, a predicted unit, and a remotely predicted unit, such that the data units in the

20     series are organized into segments, and each segment has an independent data unit, two or more predicted units and two or more remotely predicted units, wherein the independent data unit is a data recovery point and a random access point in the series of data units, and the remotely predicted units are data recovery points in the series of data units that are classified independently from the random access point and are located closer together in the series of data units than the independent data units;

25     encoding each of the data units classified as an independent data unit in a compressed format using only information from the data unit;

encoding each of the data units classified as a predicted unit in a compressed format by encoding differences between the data unit and an adjacent data unit in the series;

encoding each of the data units classified as a remotely predicted unit in a compressed format by

30     encoding differences between the data unit and a remote, non-adjacent data unit in the segment, selected as either the independent unit or another remotely predicted unit in the segment; and

prioritizing encoded data units for transmission such that independent data units are transmitted with highest priority, remotely predicted units are transmitted with next highest priority, and predicted units are transmitted with lowest priority.
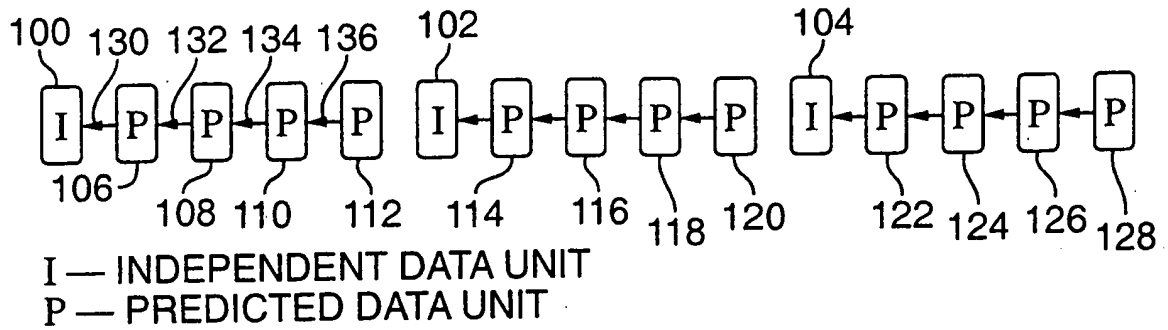
# FIG. 1

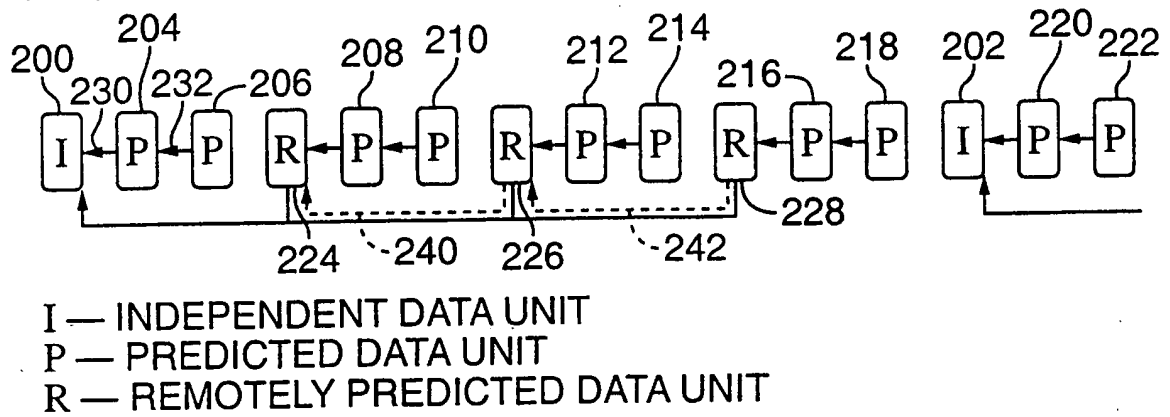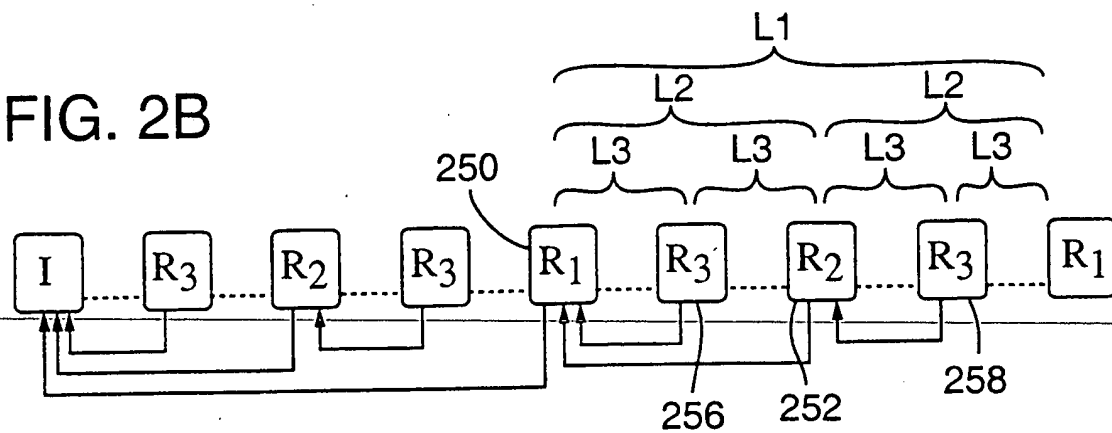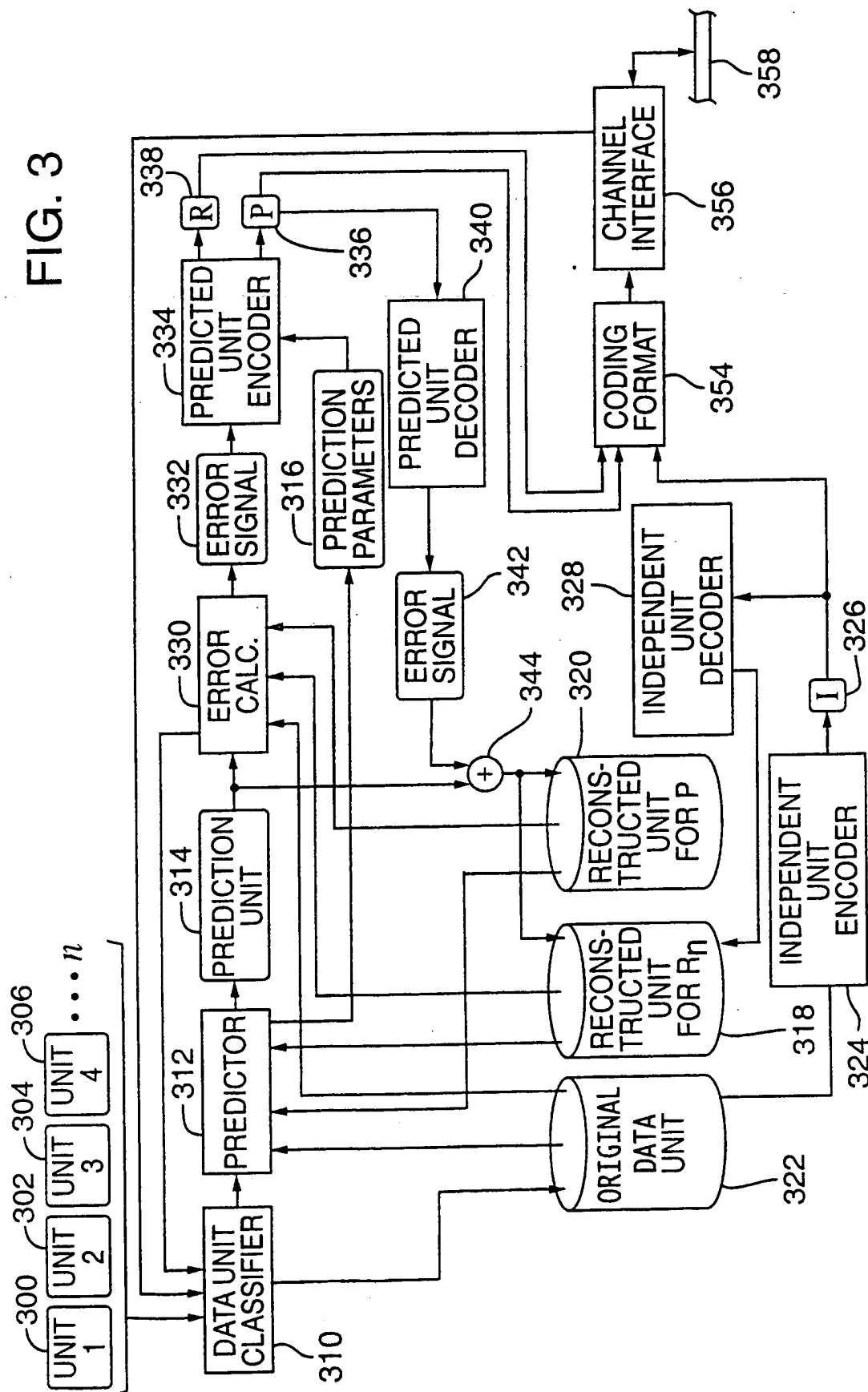I — INDEPENDENT DATA UNIT
P — PREDICTED DATA UNIT

# FIG. 2A

I — INDEPENDENT DATA UNIT
P — PREDICTED DATA UNIT
R — REMOTELY PREDICTED DATA UNIT

# FIG. 2B

FIG. 3

FIG. 4

# FIG. 5



# FIG. 6

# FIG. 7



PERSONAL COMPUTER — 720

PROCESSING UNIT — 721

723

SYSTEM MEMORY — 722

725

RAM

ROM — 724

OPERATING SYSTEM — 735

APPLICATIONS — 736

MODULES

DATA — 737

738

732 INTERFACE → HARD DRIVE — 727

733 INTERFACE → FLOPPY DRIVE — 728

DISK — 729

734 INTERFACE → CD-ROM DRIVE — 730

DISK — 731

748 VIDEO ADAPTER

MONITOR — 747

KEYBOARD — 740

MOUSE — 742

746 SERIAL PORT INTERFACE

754 MODEM

752 WAN

751

749 REMOTE COMPUTER

753 NETWORK ADAPTER

751 LAN

MEMORY STORAGE

750
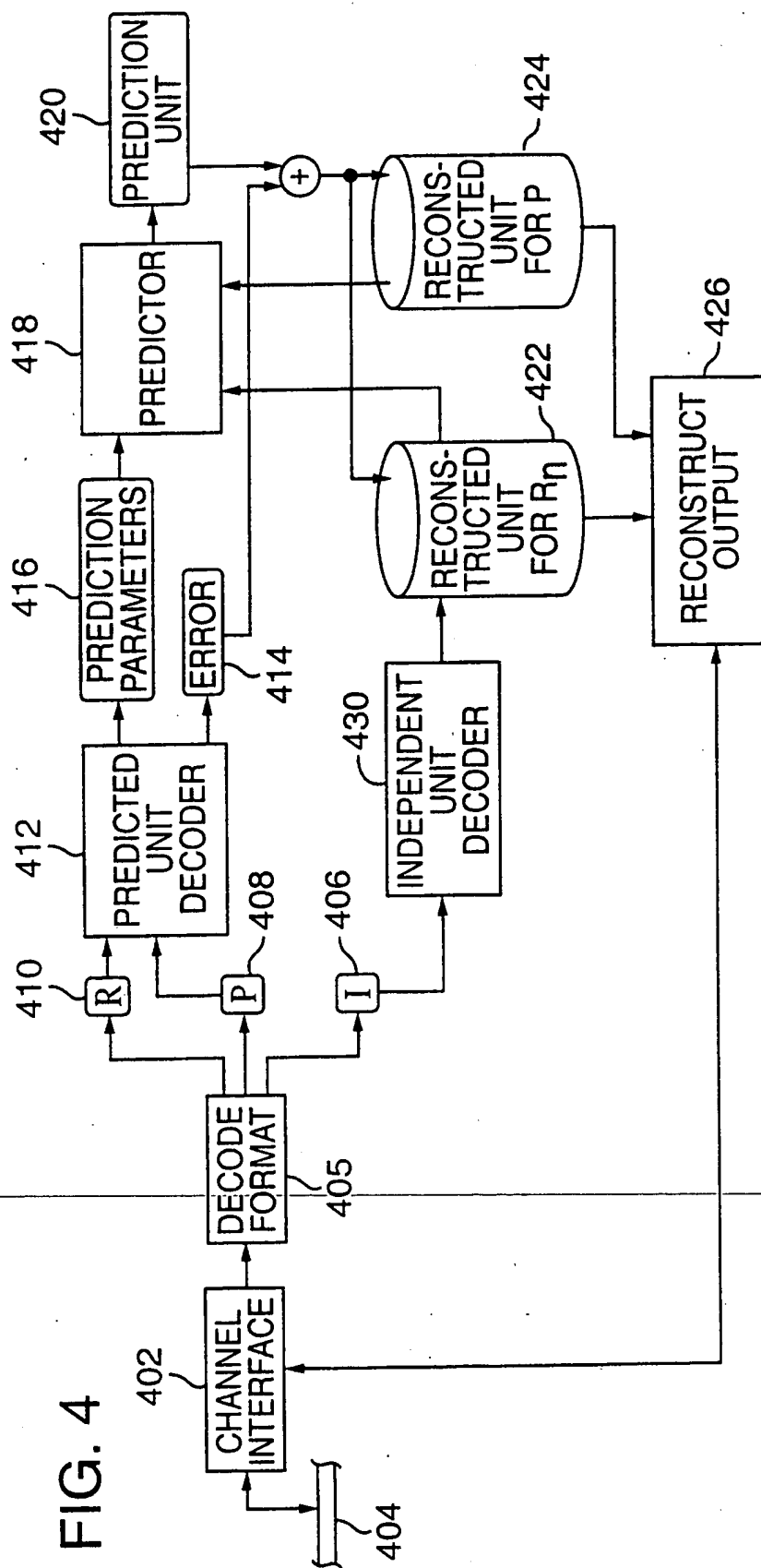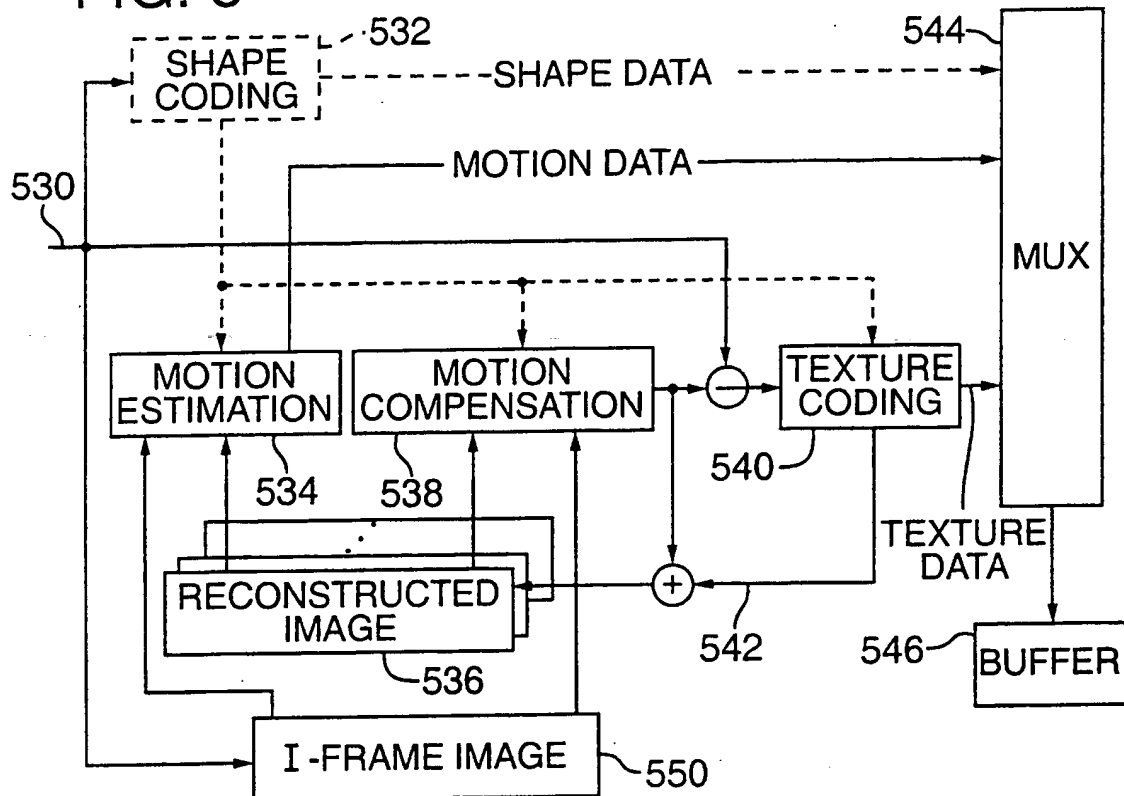
# INTERNATIONAL SEARCH REPORT

**A. CLASSIFICATION OF SUBJECT MATTER**
IPC 7   H04N7/50

According to International Patent Classification (IPC) or to both national classification and IPC

**B. FIELDS SEARCHED**

Minimum documentation searched (classification system followed by classification symbols)
IPC 7   H04N

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

EPO-Internal, INSPEC

**C. DOCUMENTS CONSIDERED TO BE RELEVANT**

| Category ° | Citation of document, with indication, where appropriate, of the relevant passages | Relevant to claim No. |
|---|---|---|
| X | US 5 592 226 A (DICKINSON BRADLEY W  ET AL) 7 January 1997 (1997-01-07)<br><br><br>abstract<br>column 2, line 34 -column 3, line 23<br>column 9, line 36 - line 53<br>column 10, line 15 -column 11, line 11<br>column 18, line 21 - line 31<br> figures 3,4,8A,8B<br>---<br><br>               -/-- | 1-3,<br>5-13,<br>15-18,<br>22,24-26 |

[X] Further documents are listed in the continuation of box C.  [X] Patent family members are listed in annex.

| Date of the actual completion of the international search | Date of mailing of the international search report |
|---|---|
| 27 July 2000 | 03/08/2000 |

| Name and mailing address of the ISA | Authorized officer |
|---|---|
| European Patent Office, P.B. 5818 Patentlaan 2<br>NL - 2280 HV Rijswijk<br>Tel. (+31-70) 340-2040, Tx. 31 651 epo nl,<br>Fax: (+31-70) 340-3016 | Barel-Faucheux, C |

Form PCT/ISA/210 (second sheet) (July 1992)

| C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT | | |
|---|---|---|
| Category ° | Citation of document, with indication,where appropnate, of the relevant passages | Relevant to claim No. |
| A | ANONYMOUS: "Method to Deliver Scalable Video across a Distributed Computer System"<br>IBM TECHNICAL DISCLOSURE BULLETIN,US,IBM CORP. NEW YORK,<br>vol. 5, no. 37, 1 May 1994 (1994-05-01), pages 251-256, XP002079392<br>ISSN: 0018-8689<br>the whole document<br>--- | 1-26 |
| A | US 5 049 991 A (NIIHARA TAKAMI)<br>17 September 1991 (1991-09-17)<br>abstract<br>column 5, line 33 - line 56<br>figure 3<br>--- | 1-26 |
| A | WIEGAND T ET AL: "Motion-compensating long-term memory prediction"<br>PROCEEDINGS. INTERNATIONAL CONFERENCE ON IMAGE PROCESSING (CAT. NO.97CB36144), PROCEEDINGS OF INTERNATIONAL CONFERENCE ON IMAGE PROCESSING, SANTA BARBARA, CA, USA, 26-29 OCT. 1997,<br>pages 53-56 vol.2, XP000923442<br>1997, Los Alamitos, CA, USA, IEEE Comput. Soc, USA<br>ISBN: 0-8186-8183-7<br>the whole document<br>--- | 1-26 |
| A | WIEGAND T ET AL: "Block-based hybrid video coding using motion-compensated long-term memory prediction"<br>PICTURE CODING SYMPOSIUM. PCS 97, BERLIN, GERMANY, 10-12 SEPT. 1997,<br>no. 143, pages 153-158, XP000924838<br>ITG-Fachberichte, 1997, VDE-Verlag, Germany<br>ISSN: 0341-0196<br>the whole document<br>--- | 1-26 |
| A | WIEGAND T ET AL: "Fast search for long-term-memory-motion-compensated prediction"<br>PROCEEDINGS 1998 INTERNATIONAL CONFERENCE ON IMAGE PROCESSING. ICIP98 (CAT. NO.98CB36269), PROCEEDINGS OF IPCIP'98 INTERNATIONAL CONFERENCE ON IMAGE PROCESSING, CHICAGO, IL, USA, 4-7 OCT. 1998,<br>pages 619-622 vol.3, XP000923421<br>1998, Los Alamitos, CA, USA, IEEE Comput. Soc, USA<br>ISBN: 0-8186-8821-1<br>the whole document<br>----- | 1-26 |

2

# INTERNATIONAL SEARCH REPORT

Information on patent family members

| Patent document cited in search report | | Publication date | Patent family member(s) | | Publication date |
|---|---|---|---|---|---|
| US 5592226 | A | 07-01-1997 | AU | 1729495 A | 15-08-1995 |
| | | | CA | 2182005 A | 03-08-1995 |
| | | | EP | 0741945 A | 13-11-1996 |
| | | | JP | 9509024 T | 09-09-1997 |
| | | | WO | 9520863 A | 03-08-1995 |
| | | | US | 5565920 A | 15-10-1996 |
| US 5049991 | A | 17-09-1991 | JP | 2219391 A | 31-08-1990 |
| | | | JP | 2521145 B | 31-07-1996 |